Concours interne de programmation
Télécom Paris

27 juin 2019

Contents

Problem A: Obélix **3**

Problem B: Thistle Darkwater **5**

Problem C: Bad-hash students **7**

Problem D: Tournament turn-around **9**

Problem E: You've goat me stumped **11**

Problem F: 256 shades of Gray **13**

Problem G: Battlecheap **15**

Problem setters

- Antoine Amarilli
- Bertrand Meyer

Proofreading

- Florian Brandner
- Bethany Cagnol

Problem tester

- Marc Jeanmougin

Judging system

- Pierre Senellart

Barème

Barème pour le cours INF280

Si vous êtes inscrit au cours INF280 en 2018–2019, votre solution pour chaque exercice sera évaluée : elle recevra la note maximale pour cet exercice si elle est acceptée par le juge en ligne, sinon elle recevra une note fractionnaire. Chaque exercice a le même poids dans le barème final.

Attention : vous devez soumettre votre solution, même partielle, sur le juge en ligne pour être corrigé.

Les soumissions incorrectes et le temps de soumission ne sont pas pris en compte dans la note.

Classement

Un classement de l'ensemble des participants sera établi à l'issue du concours. Il est indépendent du cours INF280. Il est basé uniquement sur les exercices pour lesquels votre code est accepté par le juge, et il suit les règles du SWERC.

1. Les participants sont classés en premier lieu par nombre décroissant d'exercices résolus.

2. Pour les participants qui ont résolu le même nombre d'exercices, les candidats sont classés par *temps de départage* croissant. Le temps de départage d'un candidat est la somme, pour chaque problème résolu par ce candidat, du temps entre le début du concours et la soumission de la première solution acceptée pour ce problème par ce candidat. Chaque soumission rejetée pour un problème finalement résolu par le candidat ajoute un malus de 20 minutes au temps de départage du candidat. Les soumissions rejetées sont sans effet si elles concernent un problème que le candidat n'a pas finalement résolu.

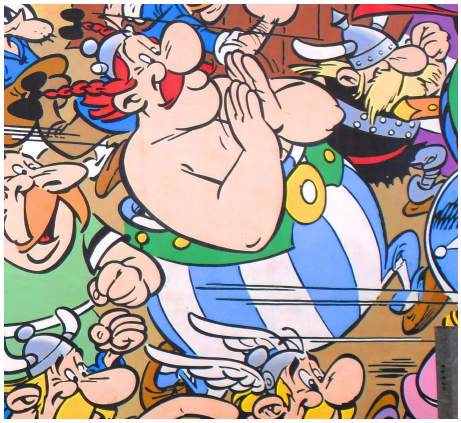
En fonction du déménagement de Télécom Paris à Palaiseau en 2019, les résultats de ce concours sont susceptibles d'être utilisés pour sélectionner les étudiants qui représenteront Télécom Paris au concours ACM-ICPC SWERC.

Problem A: Obélix

Time limit: 3 seconds

Obélix is a big eater. Everybody knows that, even the president of the Bureau des Élèves. He eats wild boar every day. But only a few people know that he is also a fine diner. In fact, Obélix never cooks his meat according to the same recipe twice. He keeps all the recipes he might want to try one day in an old book. Every recipe comes with a list of required ingredients to prepare the dish. He has even rated each recipe, in the form of a grade, according to how good he believes the dish will taste. In his kitchen pantry, he has all the required ingredients in unlimited quantities. The only drawback is that the ingredients will eventually lose their freshness and spoil as all the items of a given ingredient have the same expiration date.

In the coming days, Obélix would like to prepare a different dish every day in order to maximize the total grade of all the dishes he cooks, while never using an expired ingredient. Astérix is fortunate enough to be his guest every day, but they will have a hard time figuring out how to choose which dish to cook. Can you help?



Input

The input consists of multiple test cases. The first line of the input consists of an integer indicating the number of test cases. The first line of each test case consists of three integers n ($1 \leq n \leq 100000$), i ($1 \leq i \leq 100000$) and r ($1 \leq r \leq 100000$) separated by single spaces: n indicates the number of days during which Obélix will cook (the days are numbered from 1 to n), i is the number of ingredients (numbered from 1 to i), and r is the number of recipes in the book. The second line consists of i integers separated by single spaces: for $1 \leq j \leq i$, the j^{th} integer $1 \leq e_j \leq 100000$ indicates the day on which the ingredient will expire (i.e., the last day when it can be used). The test cases finish with r lines, one for each recipe, with the k^{th} of these lines for $1 \leq k \leq r$ consisting of integers separated by single spaces: a first integer $1 \leq g_k \leq 100$ giving the grade of the recipe, a second integer $1 \leq l_k \leq 10$ giving the number of ingredients of the recipe, and l_k distinct integers giving the required ingredients, each of which is between 1 and i.

Output

For each test case, your program should output the maximum sum of the grades of recipes that Obelix can cook over the n days, subject to the constraint that each recipe can be cooked at most once and that a recipe cannot be cooked on a day where one of its ingredients is past its expiration date.

Sample Input

```
2
2 3 2
1 2 6
5 2 2 3
10 1 1
3 3 3
1 2 3
15 1 1
5 2 2 3
10 1 1
```

Sample Output

```
15
20
```

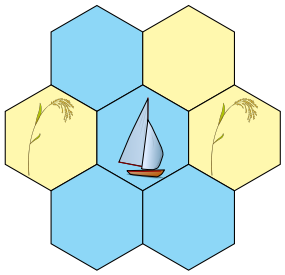
Problem B: Thistle Darkwater

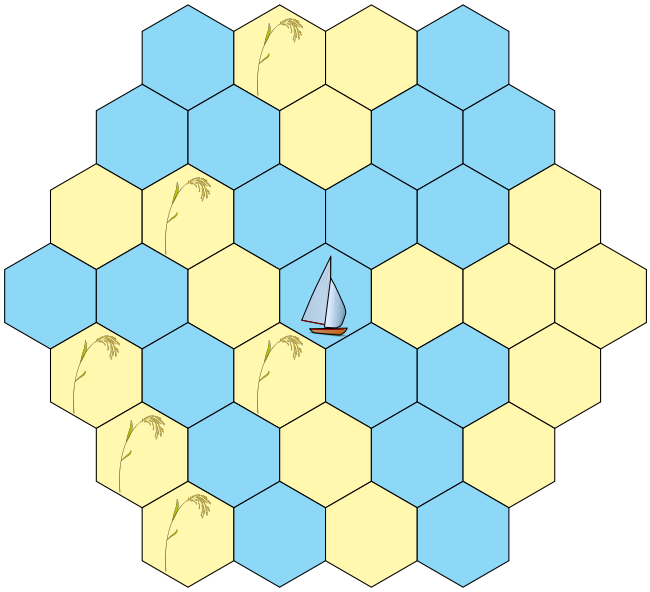
Time limit: 3 seconds

Thistle 'Silver-Eye' Darkwater is a famous explorer. After sailing a long way on the seven seas, she has reached the mysterious archipelago of the Sunken Ship Isles. The geography there is not much different from her homeland of Scotland. Islands, coves, emerged rocks, inlets, aits, lochs, holms, skerries, cays, reefs and so on. It's just a giant maze. After such a long journey, Thistle and her crew are tired. They agree to pick a land to touch down upon, fill up with fresh food and have a well-deserved break. Looking from the crow's nest, her sailors have managed to establish a map of the surrounding land formations around her ship. Thistle wants to reach a part of land that has lots of food, so that the crew's rest will be long and plentiful. Of course, she can only pick a piece of land that she can reach by sea from her current position.

Her scouts have marked the territorial situation on a map which shows water, land or food at different positions located at the points of a hexagonal lattice. What they can see forms a hexagonal region centered around the ship. Every position on the map has therefore up to six neighbors. The ship is at the center of the lattice and is always positioned on water.

Can you help Thistle to find the most plentiful piece of connected land that she can reach by sea?





Input

The input file consists of multiple test cases. The first line of the input file consists of a single integer n indicating the number of test cases. Each test case follows. The first line of a test case consists of a single integer d ($1 \leq d \leq 100$) indicating the size of the side of the hexagon that represents the territory. The next $2d-1$ lines describe the territory. Each line has $4d-3$ characters. The k^{th} line ($1 \leq k \leq 2d-1$) consists of $2d-1-|k-d|$ letters separated by single spaces and centered on the line with spaces. Each letter is either W (the cell contains water), L (the cell contains land without food), or F (the cell contains land with food). The center letter of the center line is always W.

Output

For each test case in the input, your program should produce one line consisting of one integer that indicates the maximum number of food items on a piece of connected land that Thistle's boat can reach by sea from its initial position.

Sample Input

```
2
2
 W L
F W F
 W W
4
   W F L W
  W W L W W
 L F W W W L
W W L W L L L
 F W F W W L
  F W L W L
   F W L W
```

Sample Output

```
1
2
```

Problem C: Bad-hash students

Time limit: 5 seconds

A *hash table* is a standard data structure in computer science that allows the storage of different items in a relatively small array. There are different implementations of hash tables and different ways to deal with overflows and collisions. One of which is called *hashing with open addressing and quadratic probe.*

This is how it works. For some fixed integer constant α suppose our hash table T has length n and consists of the cells $T[0]$, $T[1]$, $T[2]$, ..., $T[n-1]$. In order to save an object, we first apply an initial hash function to compute an initial key k_1, which is simply an index between 0 and $n-1$. We try to store the object at position k_1 in T. If $T[k_1]$ is already used, we try another position at index $k_2 = k_1 + \alpha \cdot 1^2 \mod n$. If $T[k_2]$ is also already used, we continue to probe the hash table at positions

$$k_i = k_1 + \alpha \cdot (i-1)^2 \mod n \text{ for } i = 3, 4, 5, \ldots,$$

and so on until we find the first empty position in the array T. If we are unable to find an empty cell of T after n probes, we give up and raise an exception. This way of storing objects is supposed to work well when the hash function that computes the initial key k_1 uniformly distributes its values among the integer interval $[\![0, n-1]\!]$ and when the number m of objects to be stored is a little smaller than n.

The sophomore students of Professor Amaretto's class have been asked to implement hash tables. Alas, not one student got it right — a horrendous situation that would never have happened in Professor Mayor's freshman class! It took a long time for Professor Amaretto to understand what his students had coded. Their first mistake was that their complicated choice of initial hash function was buggy and in fact always returns the same value k_1. Their second mistake was that they all coded the formula

$$k_i = k_1 + \alpha \cdot {k_{i-1}}^2 \mod n$$

instead of the correct formula.

Still, Professor Amaretto is curious to know how usable are these broken hash tables, implemented incorrectly by his bad-hash students.

Input

The input file consists of multiple test cases. The first line of the input file consists of a single integer c indicating the number of test cases. Each test case follows, and consists of, a single line with 3 integers n, k_1, α separated by single spaces, where n ($2 \leq n \leq 200000000$) is the size of the hash table, k_1 ($0 \leq k_1 \leq n-1$) is the constant return value of the wrongly coded initial hash function, and α ($1 \leq \alpha \leq n-1$ and $\alpha \leq 100$) is the coefficient used for the buggy quadratic probing.

Output

For each test case in the input, your program should produce one line consisting of one integer that indicates the number of cells of the hash table that can be assigned.

Sample Input

```
2
3 0 1
7 3 3
```

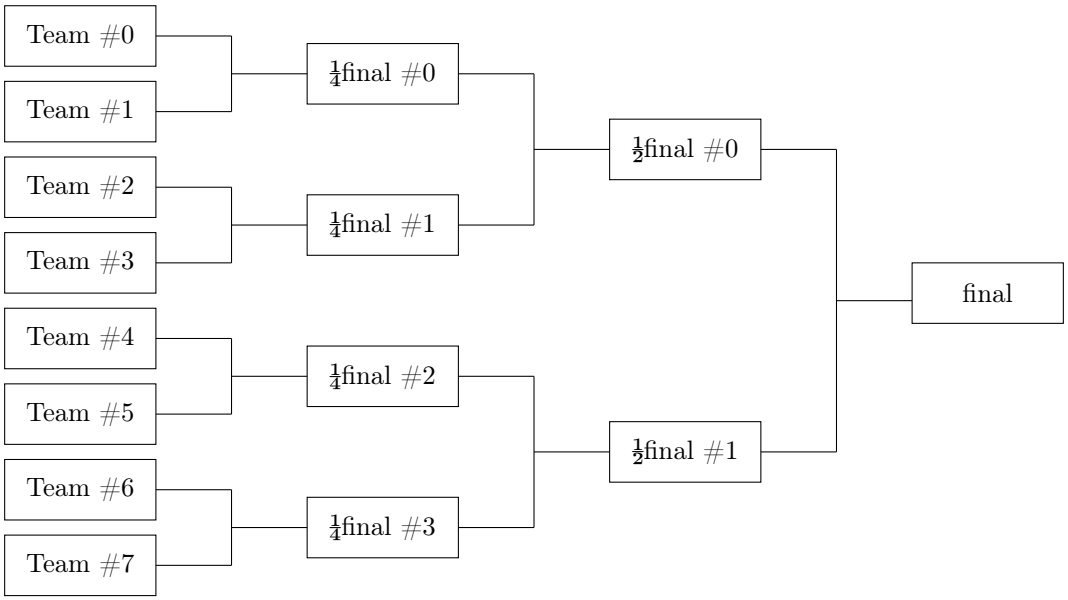
Sample Output

```
1
4
```

Problem D: Tournament turn-around

Time limit: 5 seconds

In the region of Concordia, the teams of a famous alliance of technical universities hold a wallyball tournament every year. Your university, *Beaver Valley Institute of Mines and Telecommunication Tech*, is a proud participating institution. The model of the tournament is quite simple. As there are eight teams, they are arranged in four pairs to play a quarter-final. The winners of the quarter-final are arranged in two pairs to play two semi-finals. And finally, the winners of the semi-finals meet for the final.



This year is a bit special because the alliance between the universities is about to break down. So not only will your *alma mater* lose a part of its name, which is a sport of its own at your university, but this is also your university's last chance to win the trophy that no one will ever be able to claim back. Your university has always practiced fair play and would never, ever, ever, ever cheat. But ... what if good fortune could help your team just a little bit by drawing a favorable order for the teams?

The Bureau des Sports has collected extensive statistics; over the alliance's relatively short history, you know how many matches each team has won in total over the others. We assume that this gives us the probability of the outcome of each individual match. If team p has won n times over team q, and team q has won m times over team p, then in a match between team p and q we assume that p will win with probability $\frac{n}{n+m}$ and team q will win with probability $\frac{m}{n+m}$ (there are no ties). In particular, if historically p has never won against q then you estimate it has no chance of doing so at the tournament. You have seen every pair of teams play together at least once, so the probabilities are always well-defined.

According to this probabilistic model, what would be the order of teams that maximizes the probability of your team winning the entire tournament?

Input

The input file consists of multiple test cases. The first line of the input file consists of a single integer c indicating the number of test cases. Each test case follows and consists of eight lines of

eight integers separated by single spaces. For $1 \leq i, j \leq 8$, the j^{th} integer in the i^{th} line indicates how many matches the team $i-1$ won against team $j-1$. Each integer is between zero and five, the integers in the diagonal are always zero, and it is guaranteed that every pair of teams played at least once, i.e., for $i \neq j$, either the i^{th} integer of the j^{th} line or the j^{th} integer of the i^{th} line is nonzero. Your university is always the first university in the input ordering.

Output

For each test case in the input, your program should produce one line consisting of a float, followed by eight integers separated by a single space. The float should indicate the maximum probability that your university wins the tournament, rounded to exactly 5 decimal points. The eight integers indicate how teams should be ordered to achieve this probability, assuming that the matches proceed according to the diagram above and that each individual match is decided according to the probabilistic model. If multiple arrangements give the same probability, you should output the one that is lexicographically smallest.

Sample Input

```
2
0 1 1 1 1 1 1 1
0 0 1 1 1 1 1 1
0 0 0 1 1 1 1 1
0 0 0 0 1 1 1 1
0 0 0 0 0 1 1 1
0 0 0 0 0 0 1 1
0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 0
0 4 1 2 3 0 1 1
1 0 1 5 4 3 2 5
4 4 0 4 2 1 5 5
3 5 1 0 3 5 4 4
2 1 3 2 0 5 4 3
5 2 4 0 0 0 1 4
4 3 5 1 1 4 0 2
4 5 0 1 2 1 3 0
```

Sample Output

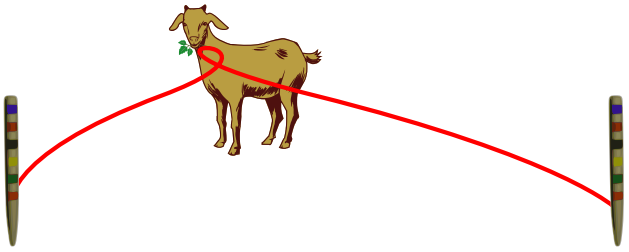
```
1.00000 0 1 2 3 4 5 6 7
0.14054 0 1 3 7 2 5 4 6
```

Problem E: You've goat me stumped

Time limit: 10 seconds

When leaving goats to graze in a field, in order to ensure that they remain in place, it is customary to drive a wooden stake in the middle of the field and tether the goat to it to ensure that it does not run away. This allows the goat to roam within a disk-shaped pasture.

In the far region of Straitmeadows, the fields are long and narrow. This odd geometry has led to an even more peculiar habit of the shepherds. They put a collar on the goat and pass a rope through it (so that the rope can move freely), and they then tie each end of the rope to a stake. This allows the goat to move along the rope but remain sufficiently close to the two stakes.



To ensure the goat doesn't starve, the shepherds need to know (very roughly) which area of grass the goat can graze in this fashion. You will assume that the goat and the stakes are 0-dimensional objects, that the rope is 1-dimensional, and that the field is a plane.

Input

The input file consists of multiple test cases. The first line of the input file consists of a single integer c indicating the number of test cases. Each test case follows and consists of a single line which consists of two integers ℓ and r separated by a single space. The integer $0 \leq \ell \leq 30$ indicates the distance between the two stakes in meters. The integer $\ell \leq r \leq 50$ indicates the length of the rope in meters.

Output

For each test case in the input, your program should produce one line consisting of a single integer which is the area of pasture that the goat can graze, rounded to the nearest hundred of square meters.

Sample Input

```
3
0 2
10 30
30 50
```

Sample Output

```
0
700
1600
```

Problem F: 256 shades of Gray

Time limit: 1 seconds

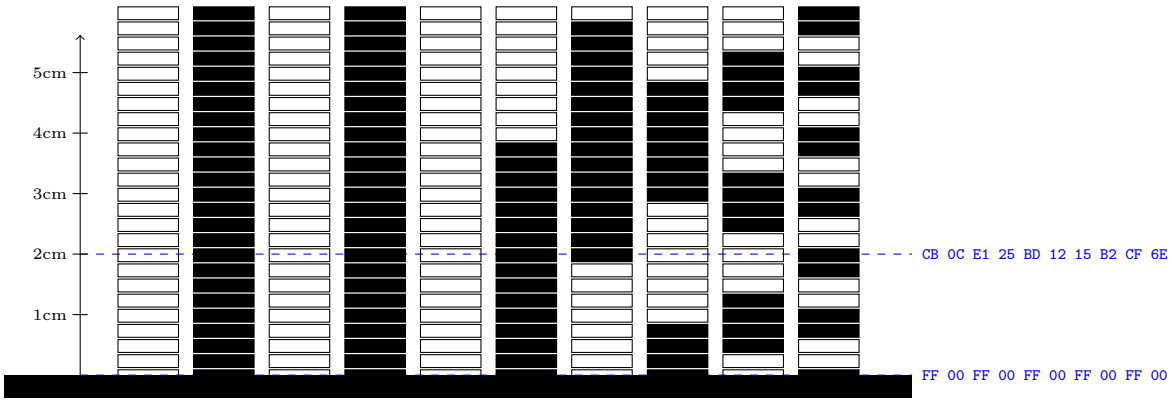
A *stadiometer* is a medical tool used to measure a person's height. In the olden days, stadiometers were just made out of a head piece rolling on a graduated ruler. But this old-fashioned technology easily led to inaccurate readings because the wheel can become decalibrated in the blink of an eye, especially with restless children.

Modern stadiometers use optical readings. They are usually able to measure your height with a precision of 0.25 cm. The way they work is that a special pattern of height 0.25 cm is printed on the wall, a laser sends a beam of light right above your head, and a photo-sensor detects the pattern that is exactly behind. There are 1024 patterns so that the device can return 1024 measurement results from 0 cm to 255.75 cm.

The International Colours & Patterns Committee has designed an open standard with the follwing rules. Each pattern consists of 10 horizontally arranged rectangles that are either black or white. From one pattern to the next, only one rectangle changes which is useful to avoid errors with the optical reading. Obviously, if the photo-sensor is positioned at the middle of two patterns and averages the readings for the two patterns, then only one rectangle can be affected and the measure is only off by 0.25 cm.

The 1024 patterns are generated in the following way:

- In the first column, the 512 bottom rectangles are white and the 512 top rectangles are black.
- For $2 \leq k \leq 10$, the k^{th} column is divided into 2^{k-2} blocks of 2^{12-k} rectangles:
  - the 2^{10-k} rectangles at the bottom and at the top of the block are white if k is odd and black if k is even;
  - the 2^{11-k} rectangles in the middle of the block are black if k is odd and white if k is even.



The optical reader reports 10 bytes (integer values between 0 and 255) which stand for 256 shades of grey. Any number between 0 and 127 is interpreted as a black rectangle whereas any number between 128 and 255 is interpreted as a white rectangle.

Given the 10 integers reported by the optical reader, can you find the correct height of the person according to the I.C.P.C. standard?

Input

The input file consists of multiple test cases. The first line of the input file consists of a single integer indicating the number of test cases. Each test case follows and consists of a single line consisting of ten pairs of characters separated by single spaces. Each character pair stands for a number between 0 and 255 written in hexadecimal notation.

Output

For each test case in the input, your program should produce one line consisting of one float with exactly two digits after the decimal point that indicates the height of the person that has been measured.

Sample Input

```
2
FF 00 FF 00 FF 00 FF 00 FF 00
CB 0C E1 25 BD 12 15 B2 CF 6E
```

Sample Output

```
0.00
2.00
```

Problem G: Battlecheap

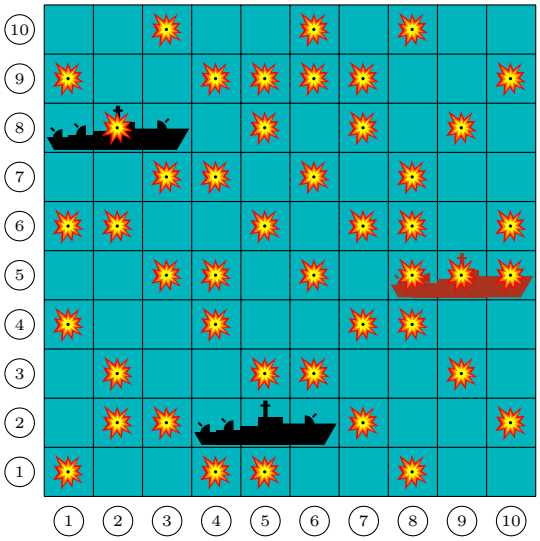
Time limit: 3 seconds

You are bored to death during a very long flight. While surfing through the collection of crappy video games provided by the in-flight entertainment system, you have stumbled upon something interesting: the game *Battleship*! Remember that this is a game for two players played on a 10 by 10 grid; your goal is to position your fleet and try to sink the enemy ships before your own are sunk.

So wouldn't it be fun to try play *Battleship* while you are flying in a plane? Unfortunately, you'll soon discover that it isn't that much fun after all and the in-flight video game is just a low-budget and buggy clone with several key shortcomings:

- You start by placing your cruiser which is 3 squares long and 1 square wide, but then the remote control doesn't seem to have the button to move to the next ship, so after a while the game automatically starts with just that ship in place.
- The computer can fire at your ships during its turn, but when it's your turn, the button to fire at the computer doesn't seem to be mapped on the remote control either. So after a while you lose your turn and the computer shoots at you again.
- Fortunately, however, the computer seems to have a limit on how many times it plays. After a certain number of shots, if you haven't lost yet (i.e. not all the squares of your cruiser have been hit), then the computer just gives up and declares you as the winner.
- As it turns out, the game's RNG is also completely buggy; from one game to the next, the computer always fires the exact same sequence of squares, no matter what happens or what it hits or where your cruiser was placed.

Of course, as soon as you have figured out the computer's hardcoded sequence of moves, this makes the game an utter disappointment. Still, you are stuck on the plane and you start to wonder, knowing this sequence, how many ways (if any) do you have to win, i.e. how many ways are there to position the cruiser and survive until the computer gives up?



Input

The input file consists of multiple test cases. The first line of the input file consists of a single integer indicating the number of test cases. Each test case follows. The first line consists of a single integer $0 \leq n \leq 100$ indicating how many times the computer fires before giving up. The next n lines describe the successive squares where the computer fires: the i^{th} line consists of two integers $1 \leq r_i \leq 10$ and $1 \leq c_i \leq 10$ giving the row and column of the computer's i^{th} shot. It is guaranteed that the computer never fires twice at the same position in its sequence.

Output

For each test case in the input, your program should produce one line consisting of one integer indicating in how many different ways you can position your cruiser at the beginning of the game and survive until the end. The cruiser must be positioned on 3 squares that are vertically adjacent or horizontally adjacent, and you survive unless the computer's sequence hits all the squares where the cruiser is placed.

Sample Input

```
2
0
10
3 2
4 7
2 2
3 6
3 3
4 9
3 4
3 7
4 8
3 5
```

Sample Output

```
160
155
```

Image rights

- Problem A: "Détail d'une peinture murale avec Astérix et Obélix, héros de la BD de Goscinny et Uderzo. Lieu : Rue de la Buanderie 33/35 (Washuisstraat 33/35), École maternelle 8/2, ville de Bruxelles, Belgique. Extrait retouché de File:Comic wall Asterix & Obelix, Goscinny and Uderzo. Brussels.jpg." Author: Ferran Cornellà. Licence: Creative Commons BY-SA, and freedom of panorama of Belgian copyright law. `https://fr.wikipedia.org/wiki/Fichier:Asterix%26Obelix_Brussels.jpg` (cropped)
- Problem B:
  - Sailboat image: public domain, https://pixabay.com/vectors/yacht-sailing-sailboat-sea-cruise-26603/
  - Rice image: DataBase Center for Life Science (DBCLS) `http://dbcls.rois.ac.jp/` on `https://commons.wikimedia.org/wiki/File:201109_rice.png`
- Problem E:
  - Stake image: `https://pixabay.com/vectors/croquet-stake-wood-game-148670/`, public domain
  - Goat image: `https://pixabay.com/vectors/goat-drawing-animal-1456762/`, public domain
- Problem G: `https://commons.wikimedia.org/wiki/File:Explosion-155624_icon.svg`, public domain